Lecture 10 – Basic mathematical structures: Posets, chains, anti-chains
Recap: Partial orders and equivalences relations

- Reflexive: $\forall a \in S, aRa$.
- Symmetric: $\forall a, b \in S$, if $aRb$ then $bRa$.
- Anti-symmetric: $\forall a, b \in S$, if $aRb$ and $bRa$, then $a = b$.
- Transitive: $\forall a, b, c \in S$, if $aRb$ and $bRc$, then $aRc$. 
Recap: Partial orders and equivalences relations

- **Reflexive**: \( \forall a \in S, \ aRa. \)
- **Symmetric**: \( \forall a, b \in S, \text{ if } aRb \text{ then } bRa. \)
- **Anti-symmetric**: \( \forall a, b \in S, \text{ if } aRb, bRa, \text{ then } a = b. \)
- **Transitive**: \( \forall a, b, c \in S, \text{ if } aRb, bRc, \text{ then } aRc. \)

<table>
<thead>
<tr>
<th></th>
<th>Reflexive</th>
<th>Transitive</th>
<th>Symmetric</th>
<th>Anti-symmetric</th>
</tr>
</thead>
<tbody>
<tr>
<td>Equivalence relation</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>Partial order</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td>✓</td>
</tr>
</tbody>
</table>
Recap: Partial orders and equivalences relations

- **Reflexive**: $\forall a \in S, aRa$.
- **Symmetric**: $\forall a, b \in S$, if $aRb$ then $bRa$.
- **Anti-symmetric**: $\forall a, b \in S$, if $aRb, bRa$, then $a = b$.
- **Transitive**: $\forall a, b, c \in S$, if $aRb, bRc$, then $aRc$.

<table>
<thead>
<tr>
<th>Relation</th>
<th>Reflex</th>
<th>Anti-Symmetric</th>
<th>Transitive</th>
<th>Partial Order</th>
</tr>
</thead>
<tbody>
<tr>
<td>${(a, b) \mid a, b \in \mathbb{Z}, a \leq b}$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>${(A, B) \mid A, B \in \mathcal{P}(S), A \subseteq B}$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>${(a, b) \mid a, b \in \mathbb{Z}, a &lt; b}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${(a, b) \mid a, b \in \mathbb{Z}^+, a \mid b}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>${((a, b), (c, d)) \mid (a, b), (c, d) \in \mathbb{Z}^+ \times \mathbb{Z}^+, a &lt; c \text{ or } (a = c, b \leq d)}$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Recap: Partial orders

<table>
<thead>
<tr>
<th></th>
<th>Refl.</th>
<th>Anti-Sym</th>
<th>Trans.</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>{ (a, b) \mid a, b \in \mathbb{Z}, a \leq b }</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{ (A, B) \mid A, B \in \mathcal{P}(S), A \subseteq B }</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{ (a, b) \mid a, b \in \mathbb{Z}, a &lt; b }</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ (a, b) \mid a, b \in \mathbb{Z}^+, a \mid b }</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{ ((a, b), (c, d)) \mid (a, b), (c, d) \in \mathbb{Z}^+ \times \mathbb{Z}^+, a &lt; c \text{ or } (a = c, b \leq d) }</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We use \( \leq \) to denote partial orders and write \( a \leq b \) instead of \( aRb \).
Recap: Partial orders

<table>
<thead>
<tr>
<th></th>
<th>Refl.</th>
<th>Anti-Sym</th>
<th>Trans.</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>{(a, b) \mid a, b ∈ ℤ, a ≤ b}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{(A, B) \mid A, B ∈ ℙ(S), A ⊆ B}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{(a, b) \mid a, b ∈ ℤ, a &lt; b}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{(a, b) \mid a, b ∈ ℤ⁺, a</td>
<td>b}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{((a, b), (c, d)) \mid (a, b), (c, d) ∈ ℤ⁺ × ℤ⁺, a &lt; c or (a = c, b ≤ d)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We use ≤ to denote partial orders and write a ≤ b instead of aRb.

- It is called “partial” order because, not all pairs of elements are “comparable” (i.e., related by ≤).
Recap: Partial orders

<table>
<thead>
<tr>
<th></th>
<th>Refl.</th>
<th>Anti-Sym</th>
<th>Trans.</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>{(a, b)</td>
<td>a, b ∈ Z, a ≤ b}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{(A, B)</td>
<td>A, B ∈ P(S), A ⊆ B}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{(a, b)</td>
<td>a, b ∈ Z, a &lt; b}</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{(a, b)</td>
<td>a, b ∈ Z⁺, a</td>
<td>b}</td>
<td></td>
<td></td>
</tr>
<tr>
<td>{((a, b), (c, d))</td>
<td>(a, b), (c, d) ∈ Z⁺ × Z⁺, a &lt; c or (a = c, b ≤ d)}</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We use ≤ to denote partial orders and write a ≤ b instead of aRb.
- It is called “partial” order because, not all pairs of elements are “comparable” (i.e., related by ≤).
- A **total order** is a partial order ≤ on S in which every pair of elements is comparable
  - i.e., ∀a, b ∈ S, either a ≤ b or b ≤ a.
Recap: Partial orders

<table>
<thead>
<tr>
<th>$(a, b) \mid a, b \in \mathbb{Z}, a \leq b$</th>
<th>Refl.</th>
<th>Anti-Sym</th>
<th>Trans.</th>
<th>PO</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(A, B) \mid A, B \in \mathcal{P}(S), A \subseteq B$</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>$(a, b) \mid a, b \in \mathbb{Z}, a &lt; b$</td>
<td>✓</td>
<td>✓</td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(a, b) \mid a, b \in \mathbb{Z}^+, a \mid b$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>$(((a, b), (c, d)) \mid (a, b), (c, d) \in \mathbb{Z}^+ \times \mathbb{Z}^+, a &lt; c \text{ or } (a = c, b \leq d)$</td>
<td>✓</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We use $\leq$ to denote partial orders and write $a \leq b$ instead of $aRb$.
- It is called “partial” order because, not all pairs of elements are “comparable” (i.e., related by $\leq$).
- A total order is a partial order $\leq$ on $S$ in which every pair of elements is comparable
  - i.e., $\forall a, b \in S$, either $a \leq b$ or $b \leq a$. 
Recap: Partial orders

<table>
<thead>
<tr>
<th>Partial order</th>
<th>Refl.</th>
<th>Anti-Sym</th>
<th>Trans.</th>
<th>PO</th>
<th>TO</th>
</tr>
</thead>
<tbody>
<tr>
<td>{(a, b) \mid a, b \in \mathbb{Z}, a \leq b}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
</tr>
<tr>
<td>{(A, B) \mid A, B \in \mathcal{P}(S), A \subseteq B}</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>×</td>
</tr>
<tr>
<td>{(a, b) \mid a, b \in \mathbb{Z}, a &lt; b}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{(a, b) \mid a, b \in \mathbb{Z}^+, a \mid b}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>{((a, b), (c, d)) \mid (a, b), (c, d) \in \mathbb{Z}^+ \times \mathbb{Z}^+, a &lt; c \text{ or } (a = c, b \leq d)}</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

- We use \( \leq \) to denote partial orders and write \( a \leq b \) instead of \( aRb \).
- It is called “partial” order because, not all pairs of elements are “comparable” (i.e., related by \( \leq \)).
- A **total order** is a partial order \( \leq \) on \( S \) in which every pair of elements is comparable
  - i.e., \( \forall a, b \in S \), either \( a \leq b \) or \( b \leq a \).
Partially ordered sets (Posets)

Definition
A set $S$ together with a partial order $\leq$ on $S$, is called a partially-ordered set or poset, denoted $(S, \leq)$. 
### Partially ordered sets (Posets)

#### Definition

A set \( S \) together with a partial order \( \leq \) on \( S \), is called a **partially-ordered set** or **poset**, denoted \((S, \leq)\).

#### Examples

- \((\mathbb{Z}, \leq)\): integers with the usual less than or equal to relation.
- \((\mathcal{P}(S), \subseteq)\): powerset of any set with the subset relation.
- \((\mathbb{Z}^+, |)\): positive integers with divisibility relation.
Graphical representation of relations: posets

Recall: any relation on a set can be represented as a graph with
▶ nodes as elements of the set and
▶ directed edges between them indicating the ordered pairs that are related.

Did these come from posets?
Do graphs defined by posets have any “special” properties?
Graphical representation of relations: posets

- Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.
- How does the graph of $(\mathcal{P}(S), \subseteq)$ look like?

Figure: Graph of a poset and its Hasse diagram

- What is “special” about these graphs?
- Graphs of posets are “acyclic” (except for self-loops).
- Starting from a node and following the directed edges (except self-loops), one can’t come back to the same node.
- Given the Hasse diagram of a poset, its reflexive transitive closure gives back the graph of the poset.
Graphical representation of relations: posets

- Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.
- How does the graph of $(\mathcal{P}(S), \subseteq)$ look like?
Graphical representation of relations: posets

- Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.
- How does the graph of $(\mathcal{P}(S), \subseteq)$ look like?

![Graph of a poset and its Hasse diagram]

Figure: Graph of a poset and its Hasse diagram

- What is "special" about these graphs?
- Graphs of posets are "acyclic" (except for self-loops).
- Starting from a node and following the directed edges (except self-loops), one can’t come back to the same node.
- Given the Hasse diagram of a poset, its reflexive transitive closure gives back the graph of the poset.
Graphical representation of relations: posets

- Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.

![Diagram of a poset and its Hasse diagram]

**Figure:** Graph of a poset and its Hasse diagram

- What is “special” about these graphs?
Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.

What is “special” about these graphs?
- Graphs of posets are “acyclic” (except for self-loops).
- Starting from a node and following the directed edges (except self-loops), one can’t come back to the same node.
Graphical representation of relations: posets

Let $S = \{1, 2, 3\}$. Recall the poset $(\mathcal{P}(S), \subseteq)$.

![Graph of a poset and its Hasse diagram](image)

**Figure:** Graph of a poset and its Hasse diagram

What is “special” about these graphs?

Graphs of posets are “acyclic” (except for self-loops).

Starting from a node and following the directed edges (except self-loops), one can’t come back to the same node.

Given the Hasse diagram of a poset, its reflexive transitive closure gives back the graph of the poset.
### Chains and Anti-chains

**Definition**

Let \((S, \leq)\) be a poset. A subset \(B \subseteq S\) is called

- a **chain** if every pair of elements in \(B\) is related by \(\leq\).

- That is, \(\forall a, b \in B\), we have \(a \leq b\) or \(b \leq a\) (or both).
Chains and Anti-chains

Definition

Let \((S, \preceq)\) be a poset. A subset \(B \subseteq S\) is called
- a **chain** if every pair of elements in \(B\) is related by \(\preceq\).
- That is, \(\forall a, b \in B, \ a \preceq b\) or \(b \preceq a\) (or both).
- Thus, \(\preceq\) is a total order on \(B\).
Chains and Anti-chains

Definition

Let $\langle S, \leq \rangle$ be a poset. A subset $B \subseteq S$ is called

- a **chain** if every pair of elements in $B$ is related by $\leq$.
- That is, $\forall a, b \in B$, we have $a \leq b$ or $b \leq a$ (or both).
- Thus, $\leq$ is a total order on $B$.

Definition

Let $\langle S, \leq \rangle$ be a poset. A subset $A \subseteq S$ is called

- an **anti-chain** if no two distinct elements of $A$ are related to each other under $\leq$. 

Chains and Anti-chains

Definition
Let \((S, \preceq)\) be a poset. A subset \(B \subseteq S\) is called

- a **chain** if every pair of elements in \(B\) is related by \(\preceq\).

- That is, \(\forall a, b \in B,\) we have \(a \preceq b\) or \(b \preceq a\) (or both).

- Thus, \(\preceq\) is a total order on \(B\).

Definition
Let \((S, \preceq)\) be a poset. A subset \(A \subseteq S\) is called

- an **anti-chain** if no two distinct elements of \(A\) are related to each other under \(\preceq\).

- That is, \(\forall a, b \in A, a \neq b,\) we have neither \(a \preceq b\) nor \(b \preceq a\).
Chains and Anti-chains: examples

- Let $S = \{1, 2, 3\}$.

Figure: Graph of poset $(\mathcal{P}(S), \subseteq)$ and its Hasse diagram

- What are the chains in this poset?
Chains and Anti-chains: examples

- Let \( S = \{1, 2, 3\} \).

Figure: Graph of poset \((\mathcal{P}(S), \subseteq)\) and its Hasse diagram

- What are the chains in this poset?
- What are the anti-chains in this poset?
Chains and Anti-chains: examples

- Let $S = \{1, 2, 3\}$.

Figure: Graph of poset $(\mathcal{P}(S), \subseteq)$ and its Hasse diagram

- What are the chains in this poset?
- What are the anti-chains in this poset?
- Give an example of an infinite chain & anti-chain in $(\mathbb{Z}^+, |)$. 
Examples and applications

A task scheduling example
Let us represent a recipe for making Chicken Biriyani as a poset!

- boil egg
- cut veg
- heat oil
- clean/cut chkn
- steam rice
- make garnish
- add spices/saute
- saute/cook
- mix
- arrange/serve

- Clearly, this shows the dependencies.
Examples and applications

A task scheduling example

Let us represent a recipe for making Chicken Biriyani as a poset!

- boil egg
- cut veg
- heat oil
- clean/cut chkn
- steam rice
- make garnish
- add spices/saute
- saute/cook
- mix
- arrange/serve

- Clearly, this shows the dependencies.
- But when you cook you need a total order, right?
Examples and applications

A task scheduling example

Let us represent a recipe for making Chicken Biryani as a poset!

- Clearly, this shows the dependencies.
- But when you cook you need a total order, right?
- Further, this total order must be consistent with the po.
- This is called a linearization or a topological sorting.
Topological sorting

Definition

A **topological sort** or a **linearization** of a poset \((S, \leq)\) is a poset \((S, \leq_t)\) with a total order \(\leq_t\) such that \(x \leq y\) implies \(x \leq_t y\).
## Topological sorting

### Definition

A **topological sort** or a **linearization** of a poset \((S, \leq)\) is a poset \((S, \leq_t)\) with a total order \(\leq_t\) such that \(x \leq y\) implies \(x \leq_t y\).

### Theorem

Every finite poset has a topological sort.
Topological sorting

Definition

A topological sort or a linearization of a poset \((S, \preceq)\) is a poset \((S, \preceq_t)\) with a total order \(\preceq_t\) such that \(x \preceq y\) implies \(x \preceq_t y\).

Theorem

Every finite poset has a topological sort.

Proof: (H.W)

- First prove the following lemma:
  - Every finite non-empty poset has at least one minimal element (\(x\) is minimal if \(\neg\exists y, y \leq x\)).
# Topological sorting

## Definition

A **topological sort** or a **linearization** of a poset \((S, \preceq)\) is a poset \((S, \preceq_t)\) with a total order \(\preceq_t\) such that \(x \preceq y\) implies \(x \preceq_t y\).

## Theorem

Every finite poset has a topological sort.

### Proof: (H.W)

1. First prove the following lemma:
   - Every finite non-empty poset has at least one minimal element (\(x\) is minimal if \(\nexists y, y \preceq x\)).
   - What about infinite posets?
2. Then, construct the chain to complete the proof.
Parallel Task Scheduling and chains

Coming back to our example,

- What if there are many cooks, i.e., parallel processors?
- How do we schedule the tasks to minimize time used?

```
boil egg cut veg heat oil clean/cut chkn steam rice
make garnish add spices/saute
saute/cook
mix
arrange/serve
```

Assume that every task takes 1 time unit.

Clearly, we still need at least 5 time units.

That is, the length of the longest chain (length of chain = no. of elements in it).

```
boil egg cut veg heat oil clean/cut chkn steam rice
make garnish add spices/saute
saute/cook
mix
arrange/serve
```
Parallel Task Scheduling and chains

Coming back to our example,

- What if there are many cooks, i.e., parallel processors?
- How do we schedule the tasks to minimize time used?

Assume that every task takes 1 time unit.
Parallel Task Scheduling and chains

Coming back to our example,

- What if there are many cooks, i.e., parallel processors?
- How do we schedule the tasks to minimize time used?

Assume that every task takes 1 time unit.
Clearly, we still need at least 5 time units.
Parallel Task Scheduling and chains

Coming back to our example,
- What if there are many cooks, i.e., parallel processors?
- How do we schedule the tasks to minimize time used?

Assume that every task takes 1 time unit.
- Clearly, we still need at least 5 time units.
- That is, the length of the longest chain (length of chain = no. of elements in it).
Parallel Task Scheduling

**Theorem**
For any poset, there is a legal parallel schedule that runs in $t$ steps, where $t$ is the length of the longest chain.